Amendments to the Specification

1. Please amend the Abstract of the Disclosure on page 13 to be as follows with changes made shown by mark-up.

~~One embodiment disclosed relates to a~~ A method of deactivating a process by a computer operating system. Threads of the process that are currently suspendable are moved to a stopped state. A process-wide deactivation operation is initiated. The process-wide deactivation operation is called by outstanding threads of the process when the outstanding threads re-enter the operating system's kernel.

2. Please replace the paragraph on page 5, lines 1-16 to be as follows with changes made shown by mark-up.

Then, the method **200** performed by the swapper proceeds as follows for all non-zombie threads. Each non-zombie thread of the process to be deactivated is selected **204**, and a determination **206** is made as to whether that thread is currently stopped or sleeping interruptibly (i.e. at a "suspension point"). If so, then the thread is moved **208** to a stopped state (which may be called TSSTOP). If not, then a further determination **210** is made as to whether the thread is sleeping on [[memory. In]] memory, in other words, whether the thread is a memory sleeper. A memory sleeper is a thread that is sleeping due to the system being too low on memory. If the thread is a memory sleeper, then a count of memory sleepers is incremented **212**. Otherwise, a further determination **214** is made as to whether the thread is interruptible and not currently running. If so, then the thread is removed **216** from the run queue and moved **218** to the stopped state (TSSTOP). The process loops back to selecting 204 non-zombie threads of the process, as [[As]] long as the swapper determines **220** that there are more non-zombie threads of the process being deactivated. When all non-zombie threads of the process have been gone through, then the swapper continues with the step **222** shown in FIG. 2B.